XES Software Telemetry Extension

XES Software Telemetry Extension

M. Leemans C. Liu

June 22, 2017

Documents: XES Software Telemetry Extension
Authors: M. Leemans, C. Liu
Email: m.leemans@tue.nl
Date: June 22, 2017
Version: 5.2

1 Introduction and Background

During the execution of software, execution data can be recorded. With the development of process mining techniques on the one hand, and the growing availability of software execution data on the other hand, a new form of software analytics comes into reach. That is, applying process mining techniques to analyze software execution data. To enable process mining for software, event logs should be capable of capturing software-specific data.

When analyzing the performance of software applications, not only timing, but also resource utilization is of importance. The *Software Telemetry* extension supports the recording of basic performance profile related resource utilizations commonly used in many software profiler tools [1]. The resource utilizations covered in this extension are: *CPU usage, thread usage and memory usage.*

Note that detailed timing information is already captured via the *Software Event* extension. Advanced resource utilization, like database requests, cache usage, network and socket usage and file I/O, are considered out of scope for this extension.

Extension definition

Name: Software Telemetry
Prefix: swtelemetry
URI: http://www.xes-standard.org/swtelemetry.xesext
XML:
```
<extension name="Software Telemetry"
prefix="swtelemetry"
uri="http://www.xes-standard.org/
swtelemetry.xesext"/>
```

[1] See for example: Valgrind (http://valgrind.org/), YourKit (https://www.yourkit.com/), gprof (https://sourceware.org/binutils/docs/gprof/), Blackfire (https://blackfire.io/), Appdynamics (https://www.appdynamics.com/), etc.

The remainder of this extension is organized as follows. In Section 2 we explain some basic terminology. In Sections 3, 4 and 5 we detail how various telemetry stats are recorded. Section 6 provides an example, and the XES Extension definition is given in Section 7. Finally, in Section 8 provides a reference glossary.

2 Terminology

In this extension, we will use some software-specific terminology. We provided a reference glossary in Section 8. As explained in the introduction, this extension supports the recording of basic performance profile related resource utilizations commonly used in many software profiler tools: *CPU usage, thread usage and memory usage*.

CPU usage When a program is executing, it consumes processor / CPU time. Most of this time is consumed in what is known as user space. When a program loops through an array, it is accumulating user CPU time. System time is the amount of time the CPU was busy executing code in kernel space. When a program executes a system call such as exec or fork, it is accumulating system CPU time. The CPU usage telemetry indicates how much of the CPU time was used. This gives a load indication for the CPU resource.

Thread usage A thread is a program's path of execution. A multi-threaded program can execute multiple code paths in parallel, and possibly concurrently. A daemon thread is a background thread, typically used for providing a particular service for other threads. The thread count telemetry gives an indication of how many execution paths are concurrently active.

Memory usage As a program is executing, various objects and arrays are instantiated and destroyed. These objects are stored in what is known as the heap memory. The heap memory is used to allocate blocks of RAM memory on demand for use by the program, and can grow during the execution of a program. The memory usage telemetry indicates how much of the RAM memory was used. This gives a load indication for the RAM memory resource. Typically, this telemetry refers to the used heap memory.

3 CPU Usage

The CPU usage telemetry indicates how much of the CPU time was used. This gives a load indication for the CPU resource. We define the following CPU

usage attributes.

Level	Key	Type	Description
event	cpuTotalUser	int	The total time in milliseconds that the CPUs spent in user space.
event	cpuTotalKernel	int	The total time in milliseconds that the CPUs spent in kernel space.
event	cpuTotalIdle	int	The total time in milliseconds that the CPUs spent idle.
event	cpuLoadUser	float	The fraction of time that the CPUs spent in user space. 1 represents 100% usage, 0 represents 0% usage.
event	cpuLoadKernel	float	The fraction of time that the CPUs spent in kernel space. 1 represents 100% usage, 0 represents 0% usage.

4 Thread Count

The thread count telemetry gives an indication of how many execution paths are concurrently active. We define the following thread count attributes.

Level	Key	Type	Description
event	threadTotal	int	The total number of active threads.
event	threadDaemon	int	The number of active daemon threads.

5 Memory Usage

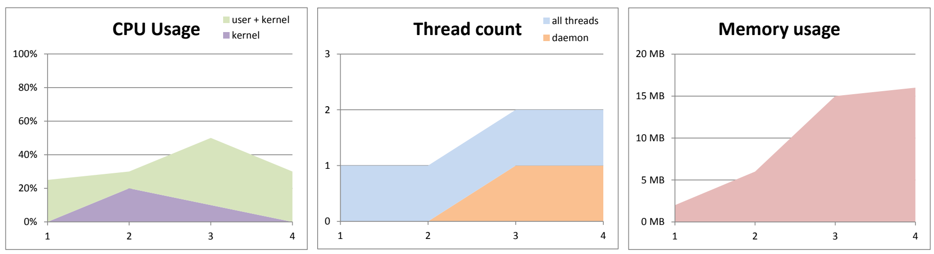
The Memory usage telemetry indicates how much of the RAM memory was used. This gives a load indication for the RAM memory resource. Typically, this telemetry refers to the used heap memory. We define the following memory usage attributes.

Level	Key	Type	Description
event	memoryUsed	int	The amount of memory used, measured in bytes.
event	memoryTotal	int	The amount of memory available, measured in bytes.
event	memoryLoad	float	The fraction of memory in use. 1 represents 100% usage, 0 represents 0% usage.

6 Example

In Listing 1 an example XES trace is given for a simple application with telemetry data. We have annotated the load attributes with how the load values

were computed based on the differences between events. For example, the *cpuLoadUser* value in the second event is calculated by taking the difference in *cpuTotalUser* divided by the total difference, i.e., user plus kernel plus idle. In Figure 1, example performance charts are given, based on the telemetrydata recorded in the example XES log in Listing 1.



(a) CPU Usage (b) Thread count (c) Memory Usage

Figure 1: Example performance charts for the telemetry data recorded in the example XES trace in Listing 1.

Listing 1: Example XES log for a simple application with telemetry data.

```
<log>
  <extension name="Concept" prefix="concept"
     uri="http://www.xes-standard.org/concept.xesext"/>
  <extension name="Software Telemetry" prefix="swtelemetry"
     uri="http://www.xes-standard.org/swtelemetry.xesext"/>
  <trace>
    <event> ----------------------------------------- Event 1
      <string key="concept:name" value="readSettings" />
      <int key="swtelemetry:cpuTotalUser" value="20" />
      <int key="swtelemetry:cpuTotalKernel" value="0" />
      <int key="swtelemetry:cpuTotalIdle" value="60" />
      <float key="swtelemetry:cpuLoadUser" value="0.25" /> ---------- 20/(20+0+60)
      <float key="swtelemetry:cpuLoadKernel" value="0" />----------- 0/(20+0+60)
      <int key="swtelemetry:threadTotal" value="1" />
      <int key="swtelemetry:threadDaemon" value="0" />
      <int key="swtelemetry:memoryUsed" value="2097152" />------------- 2 MB
      <int key="swtelemetry:memoryTotal" value="1073741824" /> --------- 1 GB
      <float key="swtelemetry:memoryLoad" value="0.00" />---------- 20480 bytes/1073741824
    </event>
    <event> ----------------------------------------- Event 2
      <string key="concept:name" value="loadFile" />
      <int key="swtelemetry:cpuTotalUser" value="37" />
      <int key="swtelemetry:cpuTotalKernel" value="34" />
      <int key="swtelemetry:cpuTotalIdle" value="159" />
      <float key="swtelemetry:cpuLoadUser" value="0.1" /> --- (37−20)/((37+34+159)−(20+0+60))
      <float key="swtelemetry:cpuLoadKernel" value="0.2" />-- (34−0)/((37+34+159)−(20+0+60))
      <int key="swtelemetry:threadTotal" value="1" />
      <int key="swtelemetry:threadDaemon" value="0" />
      <int key="swtelemetry:memoryUsed" value="6291456" />------------- 6 MB
      <int key="swtelemetry:memoryTotal" value="1073741824" /> --------- 1 GB
      <float key="swtelemetry:memoryLoad" value="0.01" />--------- 6291456 bytes/1073741824
    </event>
    <event> ----------------------------------------- Event 3
```

```
      <string key="concept:name" value="parseInput" />
      <int key="swtelemetry:cpuTotalUser" value="129" />
      <int key="swtelemetry:cpuTotalKernel" value="57" />
      <int key="swtelemetry:cpuTotalIdle" value="203" />
      <float key="swtelemetry:cpuLoadUser" value="0.4" />- - (129-37)/((129+57+203)-(37+34+159))
      <float key="swtelemetry:cpuLoadKernel" value="0.1" />- (57-34)/((129+57+203)-(37+34+159))
      <int key="swtelemetry:threadTotal" value="2" />
      <int key="swtelemetry:threadDaemon" value="1" />
      <int key="swtelemetry:memoryUsed" value="15728640" />- - - - - - - - - - - - 15 MB
      <int key="swtelemetry:memoryTotal" value="1073741824" /> - - - - - - - - - - 1 GB
      <float key="swtelemetry:memoryLoad" value="0.02" /> - - - - - - - - 15728640 bytes/1073741824
    </event>
    <event> - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - Event 4
      <string key="concept:name" value="ouputResult" />
      <int key="swtelemetry:cpuTotalUser" value="231" />
      <int key="swtelemetry:cpuTotalKernel" value="57" />
      <int key="swtelemetry:cpuTotalIdle" value="255" />
      <float key="swtelemetry:cpuLoadUser" value="0.3" /> - (231-129)/((231+57+255)-(129+57+203))
      <float key="swtelemetry:cpuLoadKernel" value="0.0" /> (57-57)/((231+57+255)-(129+57+203))
      <int key="swtelemetry:threadTotal" value="2" />
      <int key="swtelemetry:threadDaemon" value="1" />
      <int key="swtelemetry:memoryUsed" value="16777216" />- - - - - - - - - - - - 16 MB
      <int key="swtelemetry:memoryTotal" value="1073741824" /> - - - - - - - - - - 1 GB
      <float key="swtelemetry:memoryLoad" value="0.02" /> - - - - - - - - 16777216 bytes/1073741824
    </event>
  </trace>
</log>
```

7 XES Extension

Listing 2: XES Extension - Software Telemetry.

```
<extension name="Software Telemetry" prefix="swtelemetry"
  uri="http://www.xes-standard.org/swtelemetry.xesext"/>
  <event>
    <int key="cpuTotalUser">
      <alias mapping="EN" name="CPU usage - total time in user space, in milliseconds"/>
    </int>
    <int key="cpuTotalKernel">
      <alias mapping="EN" name="CPU usage - total time in kernel space, in milliseconds"/>
    </int>
    <int key="cpuTotalIdle">
      <alias mapping="EN" name="CPU usage - total time spent idle, in milliseconds"/>
    </int>
    <float key="cpuLoadUser">
      <alias mapping="EN" name="CPU usage - load in user space"/>
    </float>
    <float key="cpuLoadKernel">
      <alias mapping="EN" name="CPU usage - load in kernel space"/>
    </float>
    <int key="threadTotal">
      <alias mapping="EN" name="Total number of threads"/>
    </int>
    <int key="threadDaemon">
      <alias mapping="EN" name="Number of daemon threads"/>
    </int>
    <int key="memoryUsed">
      <alias mapping="EN" name="Total memory used, measured in bytes"/>
    </int>
    <int key="memoryTotal">
      <alias mapping="EN" name="Total memory available, measured in bytes"/>
    </int>
```

```
31      <float key="memoryLoad">
32        <alias mapping="EN" name="Memory usage load"/>
33      </float>
34    </event>
35  </extension>
```

8 Glossary

user CPU time User time is the amount of time the CPU was busy executing code in user space. When a program loops through an array, it is accumulating user CPU time.

system CPU time System time is the amount of time the CPU was busy executing code in kernel space. When a program executes a system call such as exec or fork, it is accumulating system CPU time.

kernel CPU time See System CPU time.

heap memory The heap memory is used to allocate blocks of RAM memory on demand for use by the program, and can grow during the execution of a program.

thread A thread is a program's path of execution. A multi-threaded program can execute multiple code paths in parallel, and possibly concurrently.

daemon thread A background thread, typically used for providing a particular service for other threads.